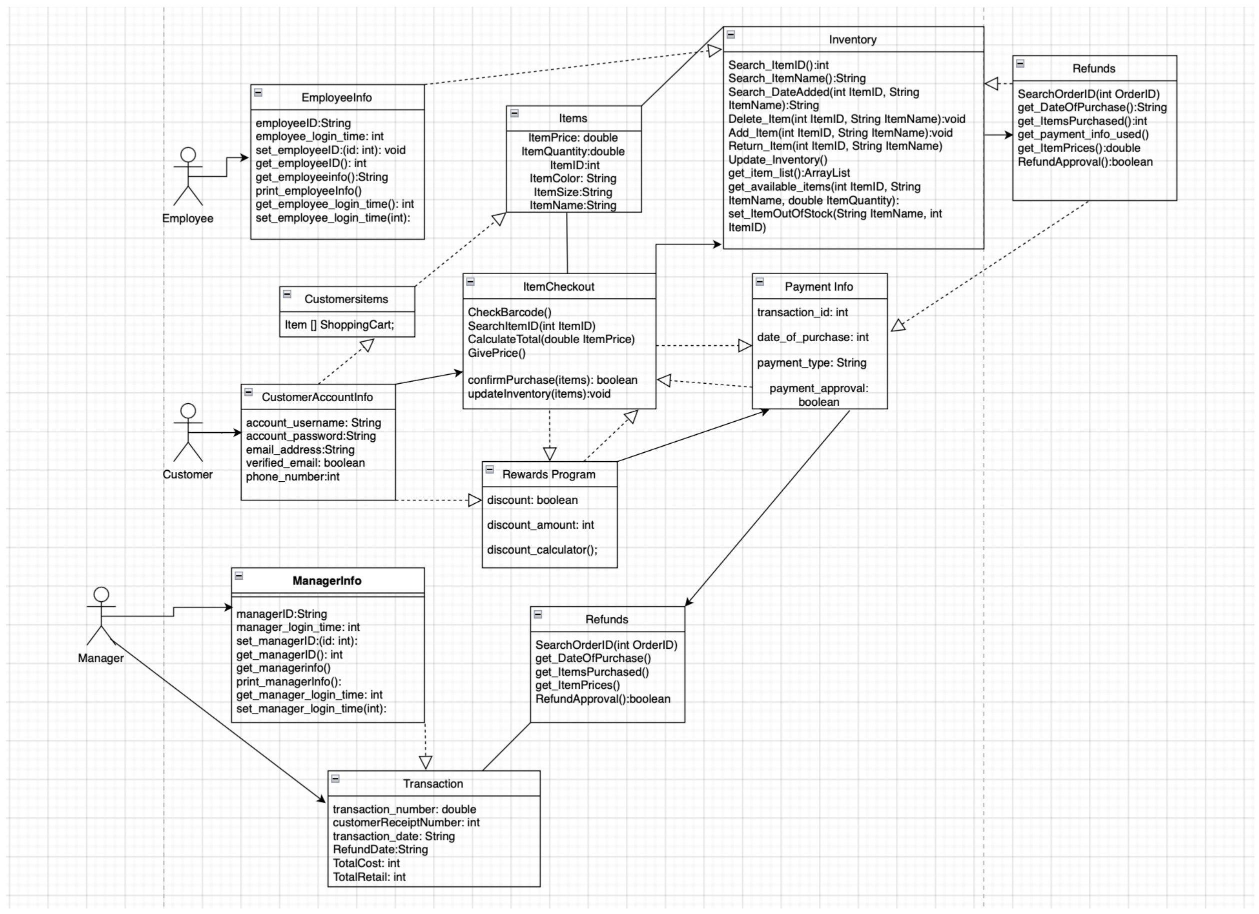
SDS: Test Plan

New Navy System

Sam Braude, Steven Alsheikh, Brandon Perillo

Updated Design Specification:

Uml Class Diagram:



UML Class Diagram Descriptions:

CustomerAccountInfo:

- Information about a customer's account. It contains their username, password, and email address, all of which are Strings. They also have a boolean to check and verify the email and a phone number as an int. The class has access for using CustomerItems and RewardsProgram, and can also be the class ItemCheckout.

<u>CustomerItems:</u>

- The inventory of a customer, only having one single attribute being Item[] ShoppingCart, which contains the items that the customer has selected to buy. The class also uses the Items class.

<u>EmployeeInfo:</u>

- The information about an employee that works in the retail system. They have an employee ID as a String, and an employee login time as an int. They have 4 methods:
    - <u>void set_employeeID(id int)</u>
        - Takes in an int and sets it as the employee ID
        - Works with int get_employeeID() so the employee ID can't be edited outside of the methods.
    - <u>int get_employeeID():</u>
        - Takes the currently set employee ID int for other uses.
        - Works with void set_employeeID(id int) so the employee ID can't be edited outside of the methods.
    - <u>void print_employeeInfo():</u>
        - Prints out the employee's ID and login time and displays them to be seen.
        - A quick method to see all of the employee's attributes.
    - <u>Int get_employee_login_time():</u>
        - Takes the employee's login time for other uses.
        - Made specifically so that it is impossible to get the login time of an employee outside of the EmployeeInfo class.

<u>ManagerInfo:</u>

- Information about a store manager, containing an ID as a String and a login time as an int. There are 4 methods:
    - <u>void set_managerID(id int):</u>
        - Takes in an int and sets it as the manager ID
        - Works with int get_managerID() so the manager ID can't be edited outside of the methods and class.
    - <u>int get_managerID():</u>
        - Takes the currently set manager ID int for other uses.
        - Works with void set_managerID(id int) so tha manager ID can't be edited outside of the methods and class.
    - <u>void print_managerInfo():</u>
        - Prints out the manager's ID and login time.
        - A quick action to see both the manager's ID and login time.
    - <u>int get_manager_login_time():</u>
        - Takes the manager's login time for other uses.
        - Is made so that the manager's login time can't be used outside of this method and the class.

<u>Items:</u>

- This class will set and display the many attributes and characteristics of the items in the system. These are clothing items such as shirts, pants, shoes, dresses, suits, etc. For this class, we have set the main attributes and characteristics that each item contains, which are the price, quantity ID, color, and size of each item.
- ItemPrice: Represented as a double, this is the amount of dollars that the customer will have to pay for a single quantity of the item.
- ItemQuantity: Represented as a double, this is the amount of items that are in stock. Customers cannot buy more items than there are in stock.

- ItemID: The specific ID of an item represented by an int. This attribute is used when items are searched for by their ID.
- ItemColor: The specific color of an item, represented by a String value.
- ItemSize: The specific size of an item, represented by a String value.
- ItemName:This stores the specific name of the item, this is represented by a String value and this will be used to help the employee search for an item through the database by providing the name of the item.

Inventory:

- This class stores, adds, removes, displays, and searches for items that are in the inventory or stock at the store. This class has methods for searching the items by using the name, ID, and date added. This class also has other methods for removing and adding methods in the inventory so the employees can see which items are in stock.
- int Search_ItemID(): This method takes in an int that represents an item id and is used for searching a specific item. This method allows the employees to search for an item id by typing in the number of that specific item. The method searches and looks through the database until it finds the specific item number.
- Search_ItemName():String: This method takes in a string and returns a string for the specific name of the item. This allows the employees to search for the item by typing in the name as a string. The method searches and looks through the database and inventory until it finds that item, and displays it on the screen for the employee.
- Search_DateAdded(int ItemID, String):String: This method takes in a string variable and searches for the date of when the item was added in the inventory. This method is responsible for searching through the database and inventory of when the item was added. The employee will need to type in the name of the item as well as the unique ID number, in order for the method to search through the database and inventory and return the date of when the item was added.

- Delete_Item(int ItemID, String ItemName):void:This method does not return anything but has a few parameters. These parameters are necessary in order for the employee to delete a specific item by typing in the name and ID number of that item. This method will be used every time an item is out of stock or not in the store.
- Add_Item():void: This method does not return anything and has no parameters. This method will be added every time the company gets more amounts of a specific item, or when a new item has been added to the inventory. This method will help the employees add the item to the database, which will then be displayed for the customers to see that they have more or new items in stock.
- Return_Item(int ItemID, String ItemName):This method takes in two parameters, the ID number of the item which is stored as an int, and the name of the item which is stored as a String. This method returns the item back to the inventory and database every time a customer gets a refund from their purchase. This will allow employees to store the item back to its original spot in the inventory and database by providing the ID number and name of the item.
- Update_Inventory(): This method does not have any parameters and does not need to return anything. This method is responsible for refreshing and updating the list of items that are in the inventory. This method will be used after the employee has added or removed items from the database. This will allow the employees to display which items are still available in the inventory.
- get_item_list():ArrayList: This method is a getter function that is stored as an ArrayList. This method is used to get the list of items that are in the inventory. This method will display all of the available items that are in the inventory, which will help the employees determine whether or not they need to remove or add any items on the list.
- get_available_items(int ItemID, String ItemName, double ItemQuantity): This method has a few parameters, it has the ID number of the item which is an int, name of the item

which is stored as a string, and the quantity of the item which is stored as a double. This method uses a getter which gets the number of available units in each item. This will help the employees determine if they are running low on stock, full in stock, or running low of items in stock.

- set_ItemOutOfStock(String ItemName, int ItemID):This method has a string name and int ID number parameter. This method is used to set a message that an item is out of stock if there are none left in the inventory. Employees will use this method to display to customers that they are out of stock for that specific item, and that it is unavailable for purchase.

<u>ItemCheckout:</u>

- This class will be used when a customer is done shopping, and is ready to get their items scanned or inserted into the cart.
- CheckBarcode(int ItemID):This method passes in the int ItemID parameter, and is used to scan and check the barcode, and make sure that the correct item name and price is displayed on the screen for the customer and employee to see. This will be stored in the customer’s receipt once they complete their purchase, so they can see which items they bought and what the prices were.
- SearchItem(int ItemID): This item passes in the int ItemID parameter, and is used to search for the item by providing in the itemID. This will search for the type of item and name of item after the unique ID number has been inputted. This will display the name of the item and the price on the screen and will later be passed on to the CalculateTotal method.
- CalculateTotal(double ItemPrice): This method takes in the double ItemPrice parameter as it will be used in calculating the total price for all the items the customer wishes to pay for. This method will calculate the total price of the items by adding in all the prices together and including the sales tax at the end. This method will also return an int for the

total price of the items, and should be displayed on screen for the customer and employee to observe.

- GivePrice(): This method will be used to display the total price for the customer to let them know how much they need to pay for all the items they wish to purchase.
- confirmPurchase(items):boolean: This method will take in a boolean value, and will be used to determine and verify that the purchase has been confirmed and that the customer is allowed to receive their items.
- updateInventory(items):void: This method will not return any values but will rather update the inventory after customers have made purchases and bought specific items. This will be needed in order to later determine if a specific item is running low or out of stock.

RewardsProgram:

- This class is used to store information about a customer's rewards or discount deals that they have and can use on certain items in the store.
- discount: This is a boolean variable that will display and determine whether or not a customer has any discount deals that they can use.
- discount_amount: This is an int variable the stores the percentage or amount that a customer gets to use as a discount to get a deal on a certain item.
- discount_calculator(): This is a method that does not have any parameters but does return an int type. This will use many mathematical operations and equations to calculate how much the customer has to pay after using many discounts and deals on items.

PaymentInfo:

- This class will store all the payment information that represents which payment type the customer used, date of purchase, transaction id, and if the payment was approved or not.

- transaction_id: This is an int variable that stores and displays a sequence of numbers generated during the electronic transfer of funds from a consumer to a merchant. The number is used to identify a transaction for recordkeeping purposes.
- date_of_purchase: This is a string variable that displays and stores the date of when the customer has made their purpose. This will also be used for recordkeeping purchases and will be checked if a customer decides to make a return on an item.
- payment_type: This is a String variable that displays the payment method the customer has decided to use. This can be cash, credit cards, debit cards, gift cards, and more.
- payment_approval: This is a boolean variable that will display whether or not a customer's payment has been approved. It will return true if the purchase has been made, and will return false if it hasn't. If false, the customer will insert the card again or try another payment method.

<u>Refunds:</u>

- This class is used to allow customers to make a refund on a purchase they made by showing the employee their receipt. This class is used to help the employee verify that the customer has made the purchase of the item that they are trying to refund by scanning the receipt. Scanning the receipt and searching the orderID number will display the items purchased and the date of purchase to the system.
- SearchOrderID(int OrderID): This method has a parameter which is int OrderID, this is an int value of the OrderID number that will allow the employee to check what items were purchased and when the purchase was made. This will search through the database to find exactly what items were purchased, time and date of purchase, location of the store at which the items were purchased, and more.
- get_DateOfPurchase():This method will be used as a getter, this will display the date of purchase of when the customer has purchased their items. The date will be stored as a string, and this will be helpful since it will allow the employee to determine if the

customer is eligible for a refund by seeing the date of purchase and checking if it has been in the past 30 days.

- get_ItemsPurchased(): This method will be used as a getter, this will display the items that the customers have purchased according to the OrderID. This will help verify whether or not the customer has purchased the item that they are trying to return.
- get_ItemPrices(): This method will be used as a getter, this will display the prices of the items that the customer has purchased according to their OrderID. This should be displayed on screen and should allow the employee to determine how much money the customer will get back if their refund has been approved.
- RefundApproval(): This method will be used to verify that the refund has been approved and processed, and that the customer will be able to receive their money back.

<u>Transaction:</u>

- This class will store all the transaction history and data, and will allow the manager and administrative users to view the translation history and data.
- Transaction_number: This is a double variable that will store the transaction number and this will be for recordkeeping purposes. Transaction IDs and numbers are located on receipts.
- customerReceiptnumber: This is an int variable that will store a customer’s receipt number so the transactions can show what the customer has purchased.
- Transaction_date: This is a string variable that will store and display the date of when a customer has made a purchase at the store or website. This will be helpful as it will help employees verify refunds by checking if a customer has purchased their returned item in the past 30 days.
- RefundDate: This is a string variable that will store and display a customer's refund date. This will be useful as managers and employees can see when they verified a refund and when they added an item back to the inventory.

- TotalCost: This is an int variable that represents the total costs that the company has made on certain items, cost of rent, salaries, profit made et.c
- TotalRetail:This is an int variable that displays and stores the total retail that the company has made, and the retail mark up price that the company has made on certain items.

Test Plan:

Brute force testing for login
Inventory: Use Brute-force for unit testing, random with integration, and partition with system testing.

Tests for Features

Feature 1: Inventory

	Input	Expected Output
Unit Test	Search for the item ID for cereal, assuming there is an item cereal. This is to test the method Search_ItemID from the class Inventory. Search_ItemID is a method that takes an int for an item ID and uses it to search and look for the item that has the specific ID.	Assuming there is a cereal item and its ID is 27625476, the method should output an integer that is 27625476.
Integration Test	Add an item with the name “banana” and an ID 48508741 and delete. This is to test the methods Add_Item and Delete_Item from the class Inventory. Both methods use Strings and ints as names and IDs respectively. Add_Item will add an item with a name and ID to stock, and Delete_Item will remove the item. Removed items	An object of the item class with the name “banana” and the ID 48508741 will be created, and then it will be “deleted”, or out of stock as the descriptions would say.

	are out of stock.	
System Test	Add an item named “FANCY SHIRT” with the ID 52249594, then search for its ID with the ItemCheckout class to purchase it. This is to test the methods for buying and checking out an item, those being the Add_Item, Search_ItemID, SearchItemID, and CheckBarcode methods, with the latter two being in the class ItemCheckout, and the rest of the methods being in the class Inventory. Add_Item uses a name for a String and ID for an int, and adds an item in stock with that name and ID. Search_ItemID is to search for the item using that ID, and SearchItemID is to get the ID of the item to use it in CheckBarcode. CheckBarcode uses the ID as an int and represents when the item is scanned.	An object of the item class with the name “FANCY SHIRT” and the ID 52249594 will be created, then its ID should be searched for and should be checked out by scanning its barcode.

Feature 2: Log in

	Input	Expected Output
Unit Test	Input the username and password for the log in. Have the get_employeeinfo() method search for the username and password, and see if it matches the username and password that was inputted in the login screen. We will use the brute-force method as there can be an infinite number of inputs for the log in.	The expected output for the unit testing would be for the system to verify that the inserted username and password matches the correct username and password for the account. If not, the system should ask the user to try again until the user has reached the limit. The system should also display an option for the user to reset their password in case they forget. The system should update the password after a user has reset it, and store that password back into the database.

Integration Test	Test valid login attempts, account lockouts, and password reset. We must prepare the necessary test data that includes the accounts, roles,and permissions. We must check to make sure the data reflects the valid usernames and passwords for the accounts. We need to test and locate any failures and errors. We must ensure that the system can identify an incorrect username and password, test to see if the account will be temporarily locked after several failed attempts, test for expired and inactive accounts and passwords.	We expect the system to identify how many attempts the user has taken for trying to type in the correct login information. The system should block the user from trying to log in to their account if they have reached the limited number of tries for trying to insert the correct login info. The system should be able to identify that a password or username has expired or if an account has been inactive.
System Test	We must test for login functionality which includes the login attempts, security features, and the overall user experience. We must also test the user interface, input validation, and security testing.	The overall system should test for login functionality and make sure the user is able to type in their account info. The system should display how many attempts the user has left for trying to insert the login info, and display other error messages that may occur during the process. The system should be able to identify and display any security issues such as brute force attacks, creation of several accounts with the same username or email, and other security issues. The system should also authorize a user access to any assets or resources in the system after the login has been successful.

(Login)

Unit Testing:

We will use brute force testing in this since there can be an excessive amount of inputs. We must verify and test the get_employeeinfo() and get_manager_info, to see if they search and get the correct account username and password. There will be multiple if-else statements in this method to check to see if they inserted username and password matches the correct one. For example, if a user enters a username such as “MikeA302” and a password like “SysLo630” the system should check to see if they match the correct username and password letter for letter and character for character. Else, the system should display an error message and allow the user to retype their login information until they have reached the limited number of attempts.

Integration Testing:

We must test the valid and invalid login attempts, test and make sure that an account will be locked after too many failed login attempts, and test to make sure that the system has successfully saved and stored a resetted password. We need to test for any errors or failures within the system, any errors or failures should be resolved. Other errors with logging in to the account should be displayed as a message to the user. We must ensure that the system can identify and verify an incorrect username and password, the system should display an error message such as "Incorrect password. Please try again." each time a user has incorrectly inserted the username and password for the account. Multiple if else statements and searches can help the system identify if a username or password is correctly inserted. The system should be able to identify if a username has been inactive or expired. We can test this by resetting a password, and going back to the login page and inserting the old password, doing so should display a message to the user saying "This password has expired. Please try again.". We also need to test for any SQL injection and other security vulnerabilities, we can test this after trying to login with multiple different accounts.

System Testing:

For the overall system, we must test for aspects of login functionality, security features and measures, and the overall user interface and user experience. We must test the login process again, to make sure the system can identify if an incorrect username and password has been inserted. We must test to see if a password has been successfully resetted or changed, so the user can log in with the new password, the new password should also be stored and saved. For the user interface, we need to test to see if any error or failed login attempt messages will be displayed, this is very important for the user so they can know if they need to enter their login information or not. We must make sure that the messages are popping up and are consistent and correct, we can't have a user make a failed password attempt and have the system display a message such as "The system had an unexpected error. Please try again". The user interface should be user-friendly and should not be too complicated for the user, there should also have information on the side addressing how a user can get help if needed. We must lastly test for security. We cannot have any account information being leaked or stolen by any one else outside of the account holder. We must test for brute force attacks, account enumeration, and other weaknesses in security. Security is the number one priority for the company, with very weak security there can be a lot of issues with a person's account information, and this will make the customer not trust the company and not purchase any products from them.